

Agile Software Development

Produced
by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>

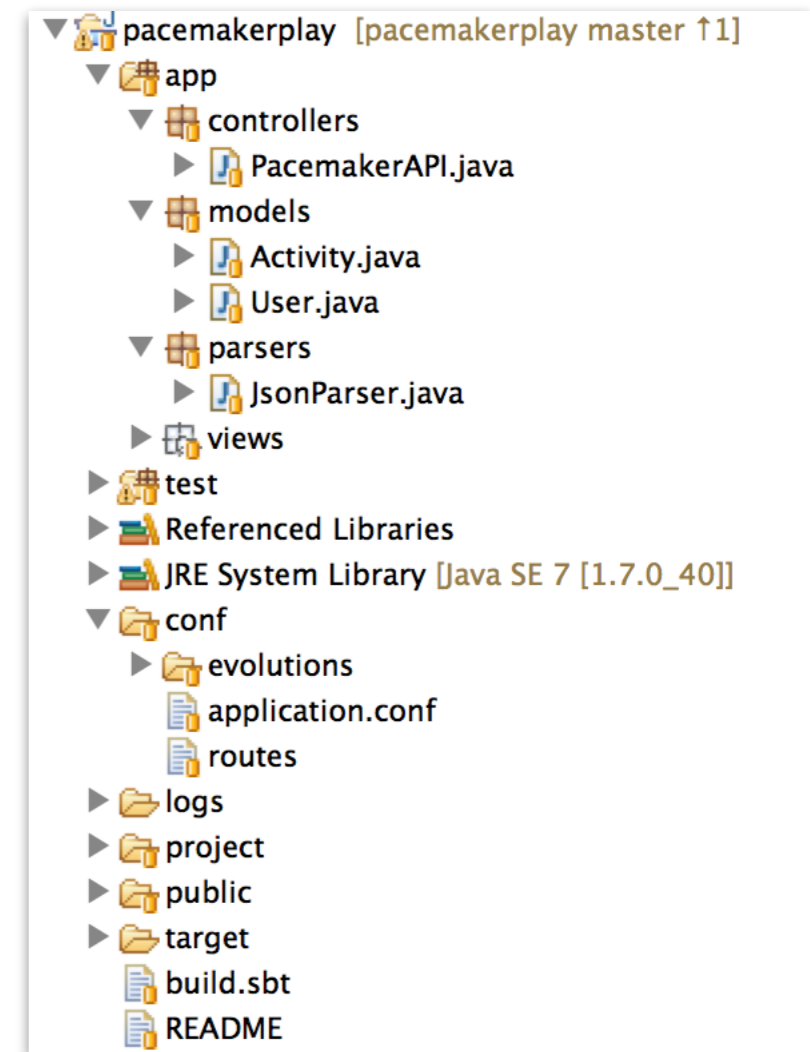
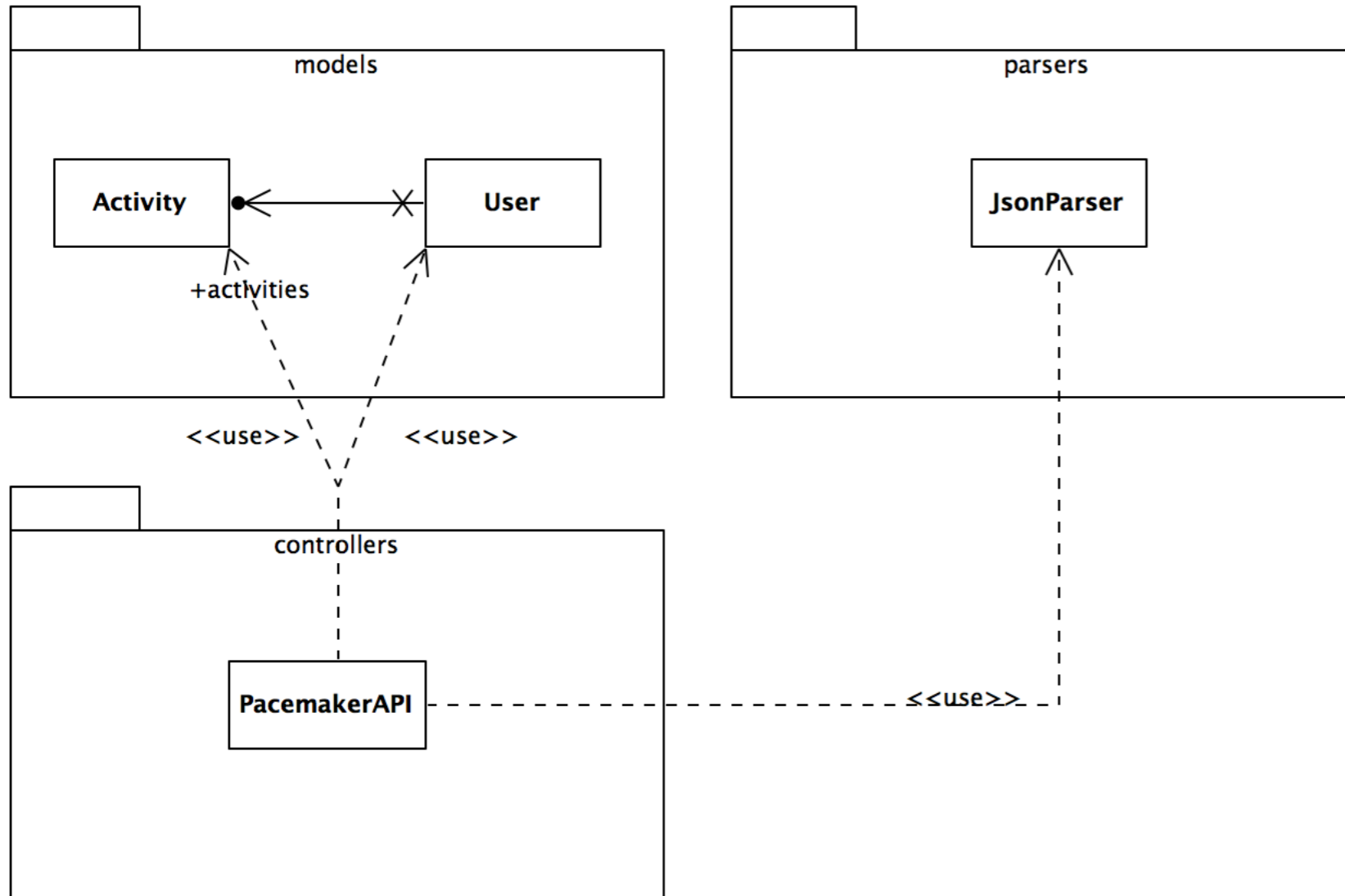


Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRCE

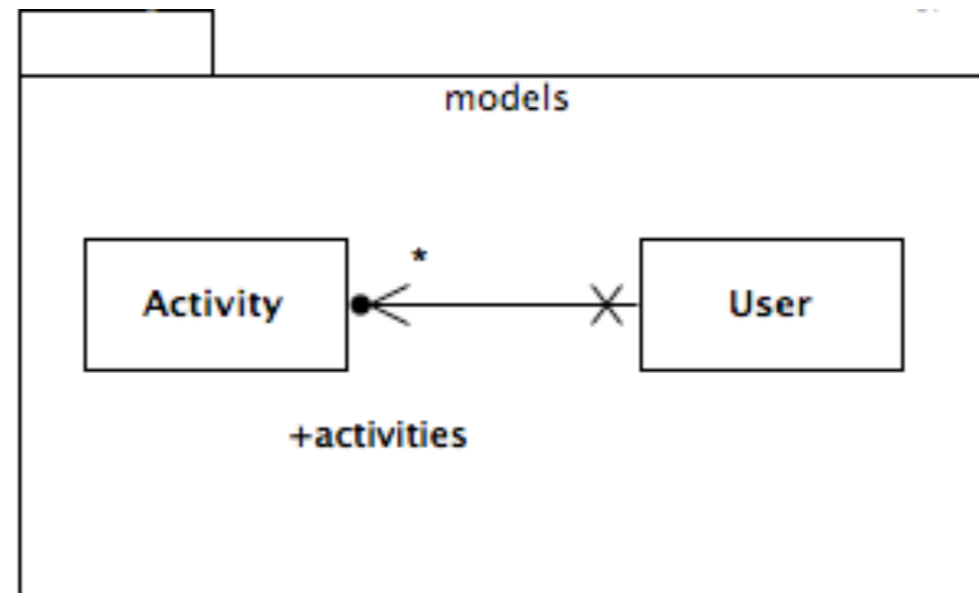


pacemakerplay v2

Model - Class/Package Diagram



Model



```
@Entity
public class User extends Model
{
    @Id
    @GeneratedValue
    public Long id;
    public String firstname;
    public String lastname;
    public String email;
    public String password;

    @OneToMany(cascade=CascadeType.ALL)
    public List<Activity> activities = new ArrayList<Activity>();

    //...
}
```

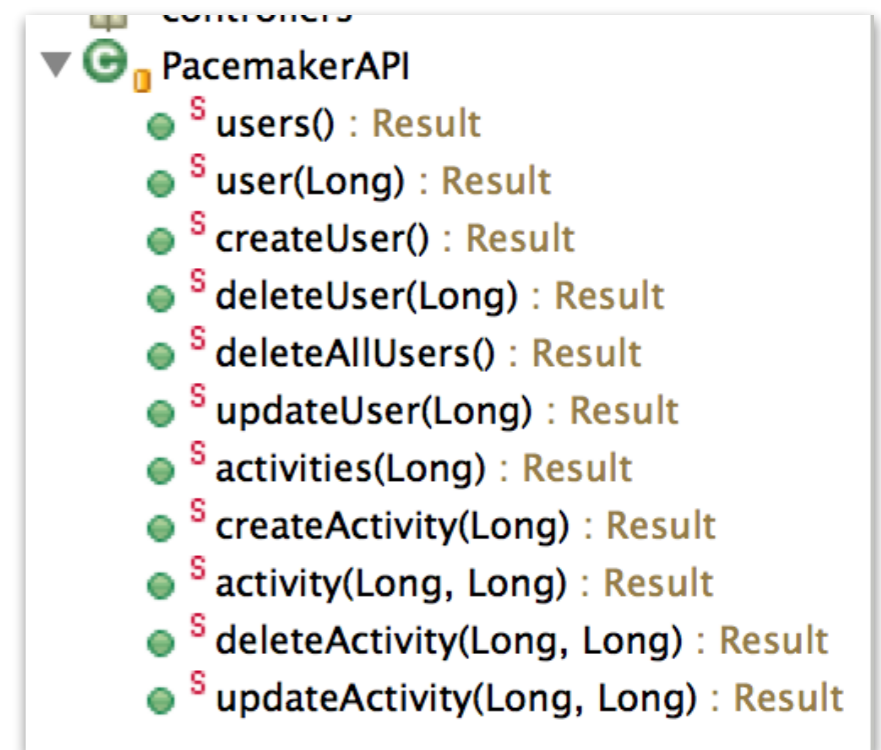
```
@Entity
public class Activity extends Model
{
    @Id
    @GeneratedValue
    public Long id;
    public String type;
    public String location;
    public double distance;

    //...
}
```

Routes + PacemakerAPI

GET	/api/users	controllers.PacemakerAPI.users()
DELETE	/api/users	controllers.PacemakerAPI.deleteAllUsers()
POST	/api/users	controllers.PacemakerAPI.createUser()
GET	/api/users/:id	controllers.PacemakerAPI.user(id: Long)
DELETE	/api/users/:id	controllers.PacemakerAPI.deleteUser(id: Long)
PUT	/api/users/:id	controllers.PacemakerAPI.updateUser(id: Long)
GET	/api/users/:userId/activities	controllers.PacemakerAPI.activities(userId: Long)
POST	/api/users/:userId/activities	controllers.PacemakerAPI.createActivity(userId: Long)
GET	/api/users/:userId/activities/:activityId	controllers.PacemakerAPI.activity(userId: Long, activityId: Long)
DELETE	/api/users/:userId/activities/:activityId	controllers.PacemakerAPI.deleteActivity(userId: Long, activityId: Long)
PUT	/api/users/:userId/activities/:activityId	controllers.PacemakerAPI.updateActivity(userId: Long, activityId: Long)

- Each route maps to a PacemakerAPI method
- Support standard Create/Read/Update/Delete operations



```
▼ G PacemakerAPI
  ● $ users() : Result
  ● $ user(Long) : Result
  ● $ createUser() : Result
  ● $ deleteUser(Long) : Result
  ● $ deleteAllUsers() : Result
  ● $ updateUser(Long) : Result
  ● $ activities(Long) : Result
  ● $ createActivity(Long) : Result
  ● $ activity(Long, Long) : Result
  ● $ deleteActivity(Long, Long) : Result
  ● $ updateActivity(Long, Long) : Result
```

API Namespace

Retrieve / Delete all users

```
GET    /api/users
DELETE /api/users
```

Create a User

```
POST   /api/users
```

Retrieve / Delete / Update specific user

```
GET     /api/users/:id
DELETE  /api/users/:id
PUT     /api/users/:id
```

Retrieve all activities for a user

```
GET     /api/users/:userId/activities
```

Create an activity for a user

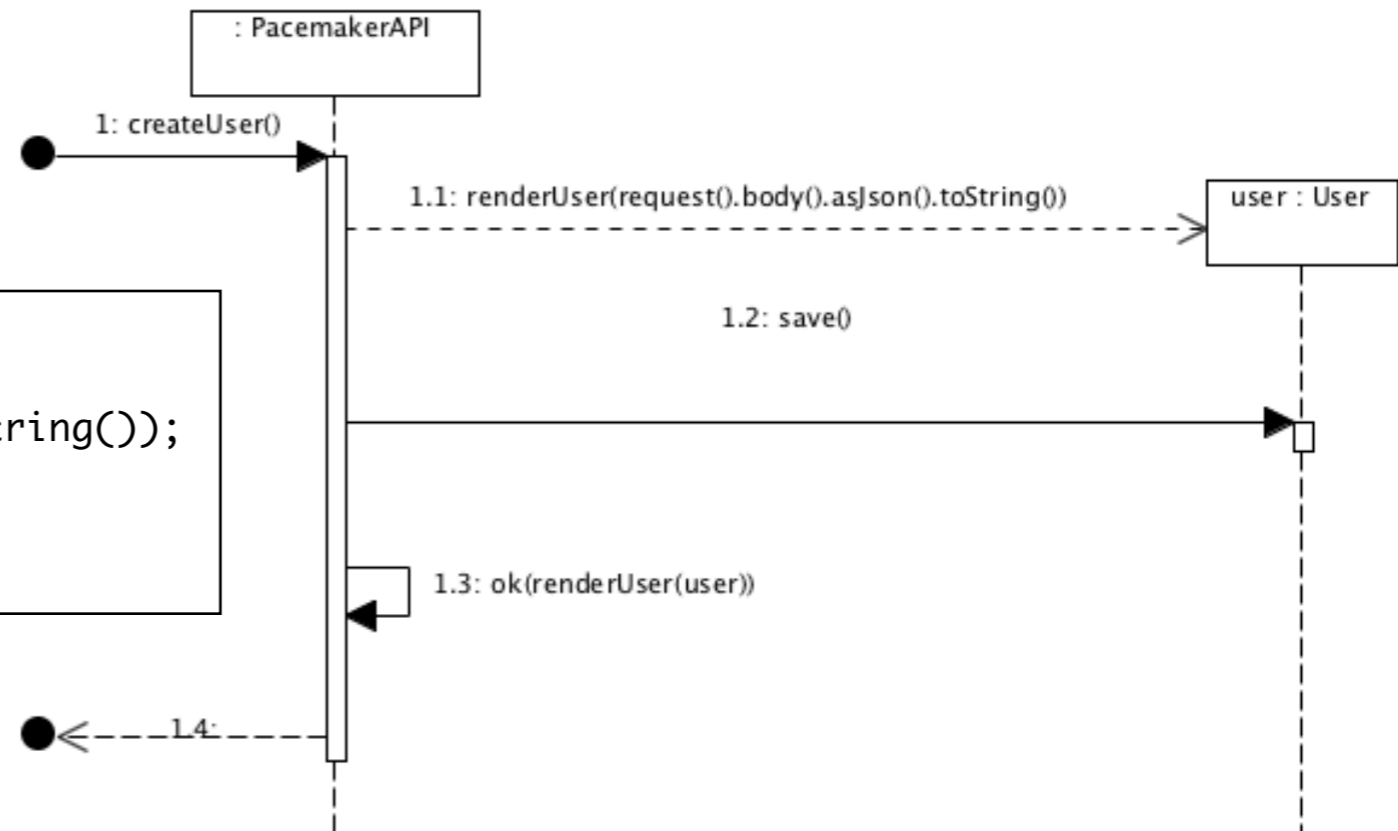
```
POST    /api/users/:userId/activities
```

Retrieve / Delete / Update specific activity

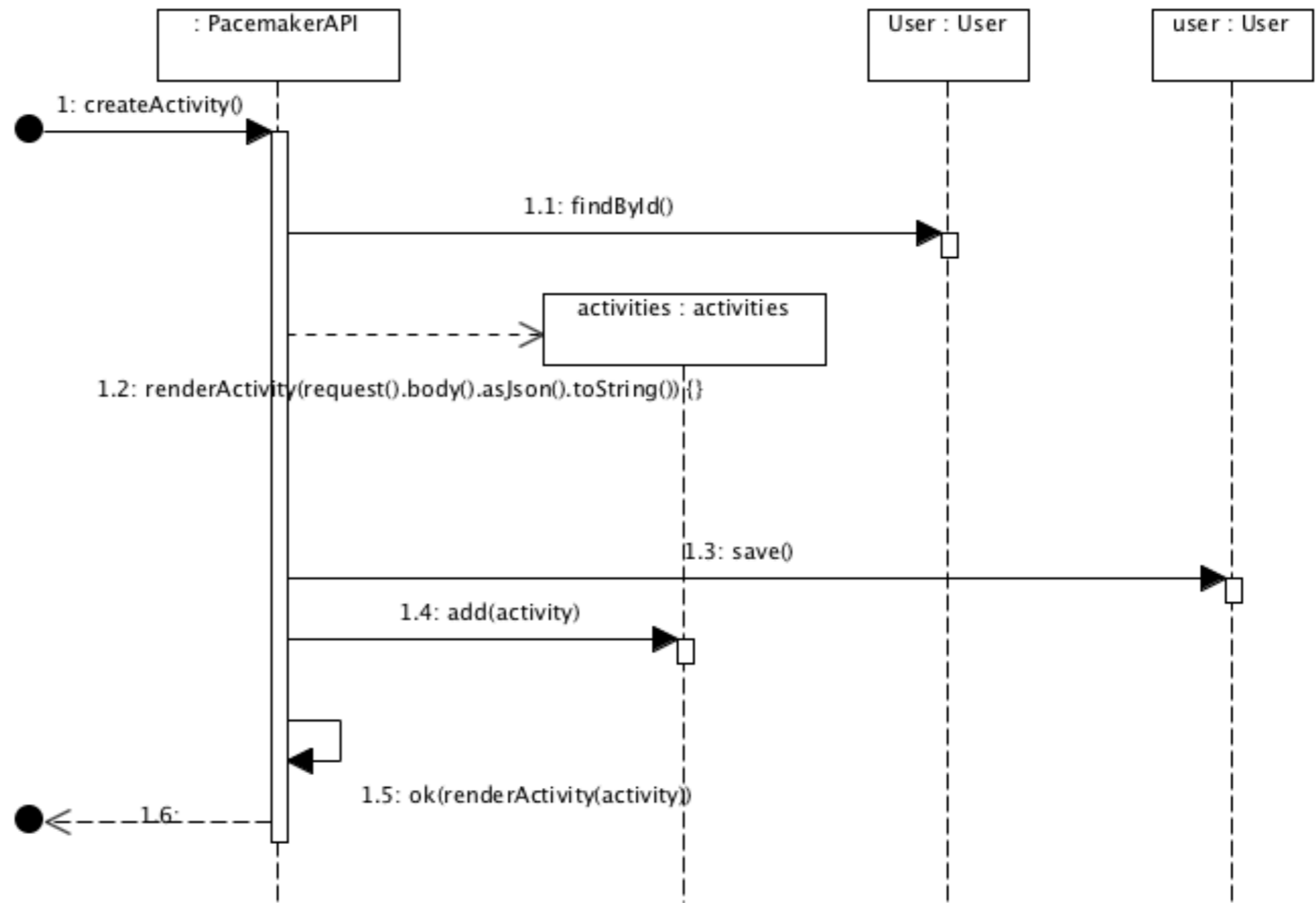
```
GET     /api/users/:userId/activities/:activityId
DELETE  /api/users/:userId/activities/:activityId
PUT     /api/users/:userId/activities/:activityId
```

createUser - Sequence Diagram

```
public static Result createUser()  
{  
    User user = renderUser(request().body().asJson().toString());  
    user.save();  
    return ok(renderUser(user));  
}
```



createActivity Sequence Diagram



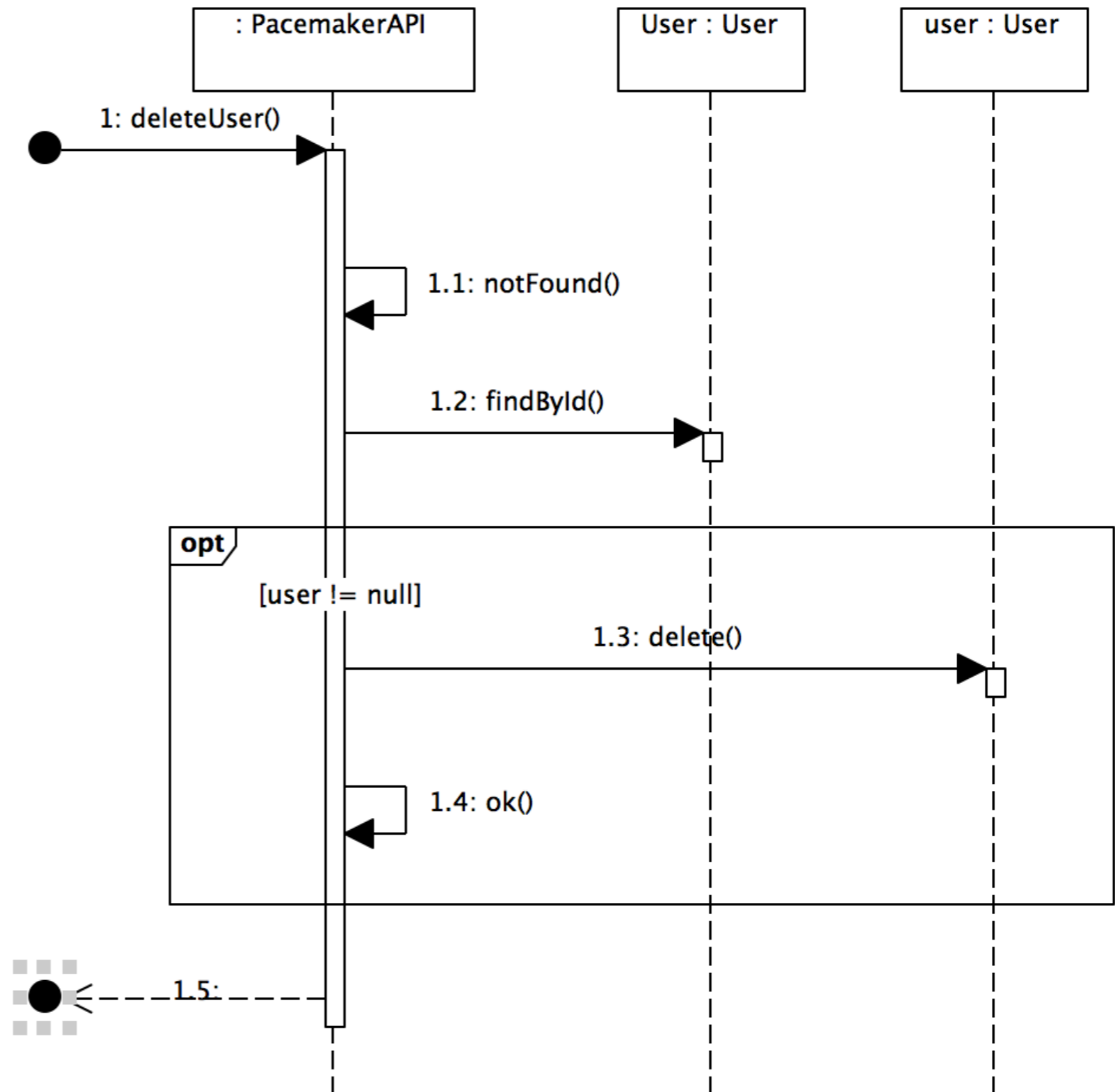
```
public static Result createActivity (Long userId)
{
    User user = User.findById(userId);
    Activity activity = renderActivity(request().body().
        asJson().toString());

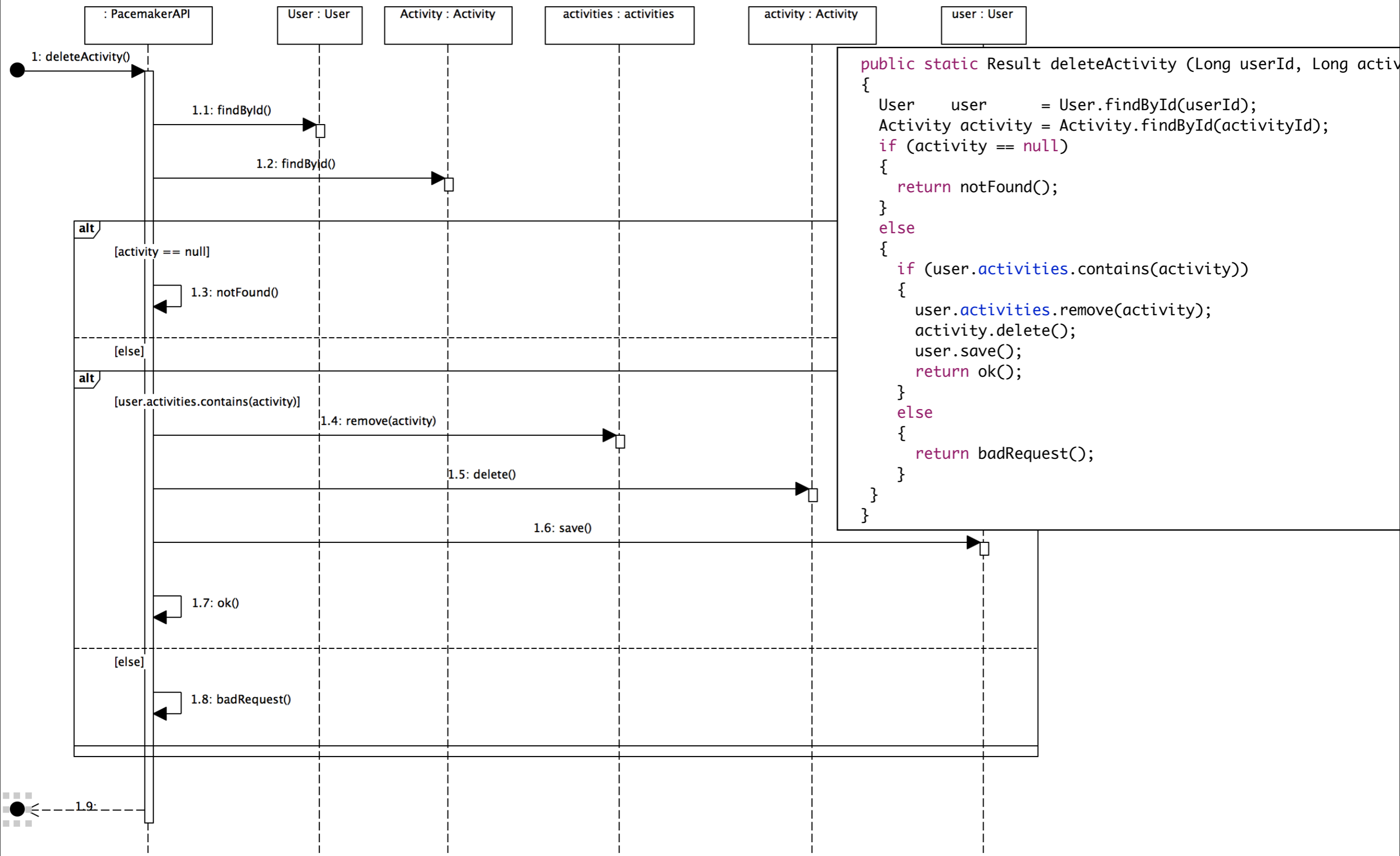
    user.activities.add(activity);
    user.save();

    return ok(renderActivity(activity));
}
```


deleteUser Sequence Diagram

```
public static Result deleteUser(Long id)
{
    Result result = notFound();
    User user = User.findById(id);
    if (user != null)
    {
        user.delete();
        result = ok();
    }
    return result;
}
```



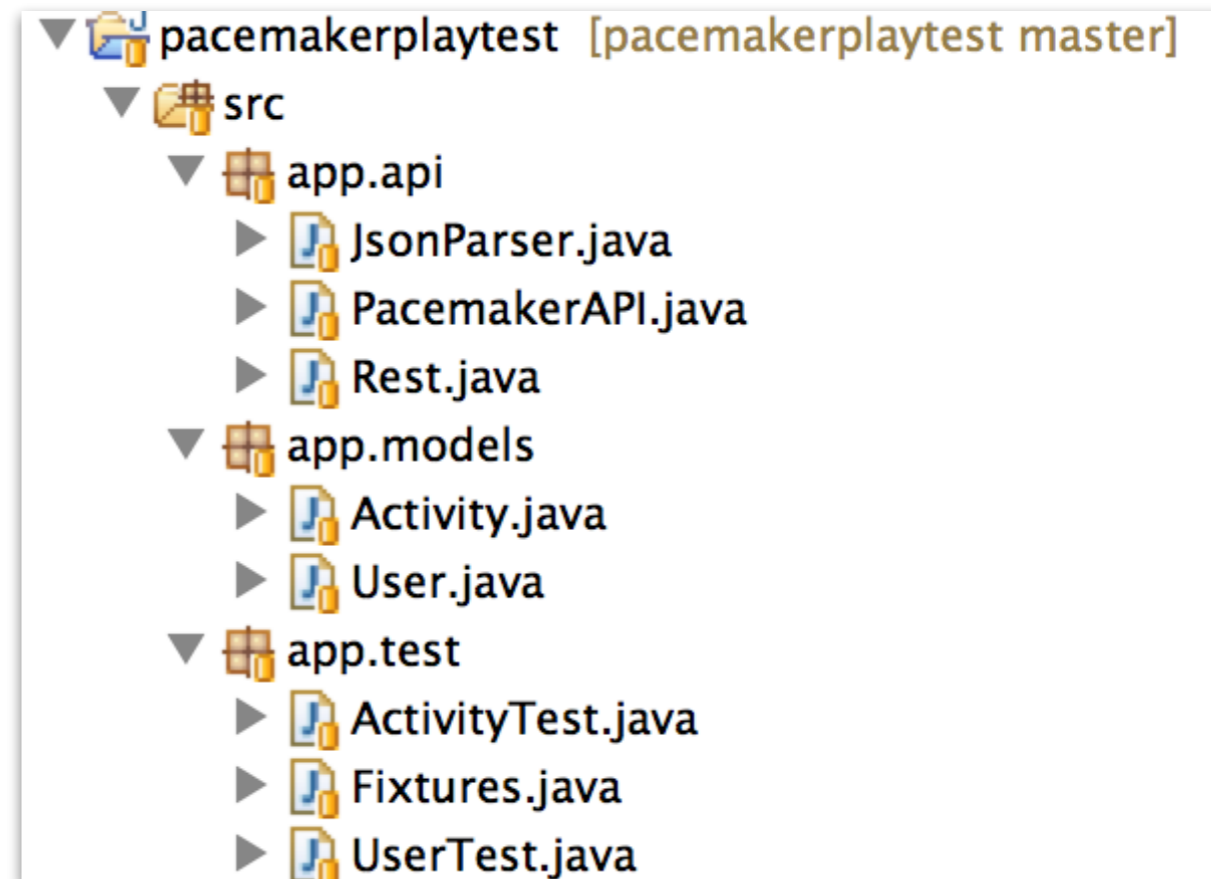


```

public static Result deleteActivity (Long userId, Long activityId)
{
    User user = User.findById(userId);
    Activity activity = Activity.findById(activityId);
    if (activity == null)
    {
        return notFound();
    }
    else
    {
        if (user.activities.contains(activity))
        {
            user.activities.remove(activity);
            activity.delete();
            user.save();
            return ok();
        }
        else
        {
            return badRequest();
        }
    }
}
  
```

deleteActivity Sequence Diagram

Pacemakerplaytest



Fixtures

```
public class Fixtures
{
    static String userJson = "{\n"
        + "\"email\"      : \"jim@simpson.com\" ,\n"
        + "\"firstName\" : \"Jim\"           ,\n"
        + "\"lastName\"  : \"Simpson\"        ,\n"
        + "\"password\"  : \"secret\"         \n"
        + "}";

    static String activityJson = "{\n"
        + "\"type\"       : \"run\"           ,\n"
        + "\"location\"   : \"Dunmore\"       ,\n"
        + "\"distance\"   : 3                 \n"
        + "}";

    static User users[] = {
        new User ("homer", "simpson", "homer@simpson.com", "secret"),
        new User ("lisa", "simpson", "lisa@simpson.com", "secret"),
        new User ("maggie", "simpson", "maggie@simpson.com", "secret"),
        new User ("bart", "simpson", "bart@simpson.com", "secret"),
        new User ("marge", "simpson", "marge@simpson.com", "secret"),
    };
}
```

ActivityTest (1)

```
public class ActivityTest
{
    private User    user;
    private Activity activity;

    @Before
    public void setUp() throws Exception
    {
        user    = new User ("mark", "simpson", "mark@simpson.com", "secret");
        activity = new Activity ("run", "tramore", 3);

        user = PacemakerAPI.createUser(Fixtures.userJson);
    }

    @After
    public void tearDown() throws Exception
    {
        Rest.delete("/api/users");
    }

    @Test
    public void createActivityJson() throws Exception
    {
        Activity activity = PacemakerAPI.createActivity(user.id, Fixtures.activityJson);
        Activity getResponse = PacemakerAPI.getActivity(user.id, activity.id);

        assertTrue(activity.equals(getResponse));

        PacemakerAPI.deleteActivity(user.id, activity.id);
    }

    @Test
    public void createActivityObj() throws Exception
    {
        Activity createResponse = PacemakerAPI.createActivity(user.id, activity);
        assertEquals(activity, createResponse);

        Activity getResponse = PacemakerAPI.getActivity(user.id, createResponse.id);
        assertEquals(activity, getResponse);

        PacemakerAPI.deleteActivity(user.id, createResponse.id);
    }
}
```

ActivityTest

```
@Test
public void updateActivity() throws Exception
{
    Activity createResponse = PacemakerAPI.createActivity(user.id, activity);
    assertEquals(activity, createResponse);

    activity.distance = 12;
    activity.location = "connemara";
    Activity updateResponse = PacemakerAPI.updateActivity(user.id, createResponse.id, activity);
    assertEquals(activity, updateResponse);

    PacemakerAPI.deleteActivity(user.id, createResponse.id);
}

@Test
public void getActivities() throws Exception
{
    Activity createResponse = PacemakerAPI.createActivity(user.id, activity);

    List<Activity> activities = PacemakerAPI.getActivities(user.id);
    assertEquals(1, activities.size());

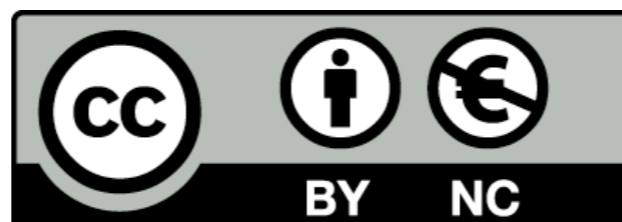
    assertEquals(activity, activities.get(0));
    PacemakerAPI.deleteActivity(user.id, createResponse.id);
}

@Test
public void getMultipleActivities() throws Exception
{
    Activity createResponse = PacemakerAPI.createActivity(user.id, activity);

    Activity activity2 = new Activity("ride", "tramore", 3);
    Activity createResponse2 = PacemakerAPI.createActivity(user.id, activity2);

    List<Activity> activities = PacemakerAPI.getActivities(user.id);
    assertEquals(2, activities.size());

    assertEquals(activity, activities.get(0));
    assertEquals(activity2, activities.get(1));
    PacemakerAPI.deleteActivity(user.id, createResponse.id);
    PacemakerAPI.deleteActivity(user.id, createResponse2.id);
}
```



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRCE

